Lab2: Problem Solving Using C++ in JGrasp - This Lab has 3 Parts

Objective - The objective of this lab is to solve a small engineering problem using C++. We will focus on these issues:

- Understand the requirements of a problem and the solution algorithm
- Implement the solution using C++ in JGrasp
  - Define floating-pt. number (non-integer) variables.
  - Take user input
  - Output results to the screen
- Debug and remove errors
- Test and verify the correctness of the results

Description- For this lab, you will calculate the area of a shaded geometric shape

Preparation- UNDERSTAND LAB 1 please.

Part 1 : Enter AND UNDERSTAND this code. Use the filename Example.cpp. This short program should give you an example of the steps you will need in the area calculation. Pay attention to the "if" statement, and the comments. Remember, everything that follows two slashes (//) is ignored by the computer. When you understand every line of this code, move on to the Area calculation in Part 2.

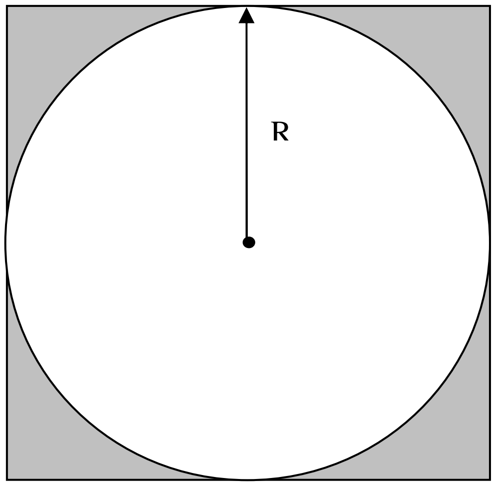
```
#include <iostream>
using namespace std;
int main( )
{
double x = 0.0;
int y = 0;

cout << "Enter a decimal number: ";
cin >> x;
cout << "The value of x is " << x << endl;

cout << "Enter an integer: ";
cin >> y;

// check if y is negative
if  ( y < 0 )
        {
        cout << "Ending program..." << endl;
        return 0;    // if so, quit the program
        }
else
        {
        cout << "The value of y is " << y << endl;
        }   // end if y < 0

} // end of program
```

… on to Part 2 …..

Part 2: Create a new C++ file called **Area2.cpp.**

Area Calculation

Your program must calculate the area of the shaded region (the square minus circle) given the radius. The radius will be input from the keyboard and validated (i.e., ensure that the radius is > 0), and the area will be calculated with decimal point ("double" number) accuracy and output. Below is the example graph:



Steps

1. *Define a "double" variable for Radius. A double is a floating-pt. number, as opposed to an integer. Initialize it to 0.0.*
2. *Define another double variable to hold the Circle's Area. Initialize it to 0.0.*
3. *Define another double variable to hold the Square's Area. Initialize it to 0.0.*
4. *Define another double variable to hold the resultant calculation of the shaded area. Initialize it to 0.0.*
5. *Define another double variable for PI and assign it the value 3.14159.*
6. *Ask the user for the Radius.*
7. *Check the validity of the radius (> 0.0). You must use an "if" command, as in the example code in part 1. If the user enters a negative radius, exit the program using **return 0;***
8. *Calculate the Circle's area using the Radius and Pi.*
9. *Calculate the Square's area using 2.0*Radius as the length of one side.*
10. *Compute and print out the area of the shaded region*
11. *Exit (return 0)*

Adding Comments

Add comments in the code (using //... or /* ... */) to explain each step. The comments are for to aid in grading your assignment, to make sure that you understand each step, so please be verbose.

Verifying Your Code - verifying the correctness of your code merely means getting correct results. Type in many different radii and make sure that the answers make sense.

If you enter a radius of -1.0, the program should reject the input.
If you enter a radius of 1.0, the result of the calculation should be: $(2*1.0)^2 - (3.14159)*1^2 = .85841$
If you enter a radius of 2.0, the result of the calculation should be:

$(2*2.0)^2 - (3.14159)*2^2 =$
$16 - (3.14159 * 4) =$
$16 - 12.56636 = 3.43364$

... on to Part 3 .....

Part 3 - Create a new file, and simply count to 1,000,000. Counting to one million is not something we would ever do by hand, but computers do it tirelessly. To accomplish this, you might take these steps:

a. Create the program structure, as always

```
#include <iostream>
using namespace std;
int main( )
{
// your program here
}
```

b. Create a counting variable. "counting" means integer. Name it something appropriate, like "loopCounter".

c. After creating and initializing your variable, create a "while" loop. The structure of a while loop is as follows:

```
while ( condition )
  {
   // do something
  }
```

in this case, the condition is (yourVariable < 1000000), and the "do something" is two lines: print out the value of your variable, use ***cout*** and don't forget the ***endl;*** and then INCREMENT your variable by 1. The line of code will be something like

```
yourVariable = yourVariable + 1;
```

d. To get your code up and running, start simply.... perhaps only count to 10, and answer these important questions:
- What should the initial value of your variable be? 0 or 1? Try each, and see what value is first displayed.
- Should the condition be "***while (yourVariable < 10)*** " or "***while (yourVariable <= 10)***" ? Try each, and see what the last displayed value is.

e. When you are satisfied that your loop boundaries are correct, replace the condition with 1000000 instead of 10, and watch your program run. How long does it take to reach 1000000?

f. Now try an experiment - move the "***cout << yourVariable << endl;***" statement from inside the loop to after the loop. In this case, the program will only print the final value when the loop is completed. NOW, how long does it take to run? You will note that the difference in execution time is due entirely to the act of printing to the screen.

g. What is the final value of your variable outside of the loop? Is it different than when the cout statement was inside the loop? Why? *the answer is: moving the cout statement to outside the while loop meant that it was printed only after the final increment. The final increment was actually 1,000,001, when the while condition failed, and then was bypassed.*

As mentioned in the syllabus, not all the labs will be graded and hence you do not need to submit this lab. We will inform you which labs need to be turned in. However, you are encouraged to complete the assignments and attend labs for a better understanding of the subject.